CANTATA
INTELLIGENT TESTING
6.2

Safety Manual

Issue 1

QA Systems GmbH
Schwieberdinger Straße 56
70435 Stuttgart
Germany

Tel +49 (0)711 / 13 81 83 - 0
Fax +49 (0)711 / 13 81 83 - 10
E-Mail info@qa-systems.de
www.qa-systems.de

QA Systems Ltd
2 Palace Yard Mews
Bath BA1 2NH
United Kingdom

Tel +44 (0) 1225 321 888
Fax +44 (0) 1225 581 150
E-mail: info@qa-systems.com
www.qa-systems.com

RACING THE DECADE WITH QA SYSTEMS

© Copyright QA Systems GmbH 2012

Executive Summary

This Safety Manual provides guidance for how to use the Cantata tool in a safety related context. It should be read in conjunction with the Cantata Manual, and the relevant Cantata standards guidance briefing paper, which maps the methods required by that safety standard to the capabilities of the Cantata tool.

This manual provides guidance for the following safety related standards:

Standard Reference	Standard Title
IEC 61508:2010	Functional safety of electrical/electronic/programmable electronic safety related systems
ISO 26262:2011	Road Vehicles – Functional Safety
EN 50128:2011	Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems
IEC 62304:2006	Medical device software – Software lifecycle processes
FDA UCM085281:2002	General Principles of Software validation; Final Guidance for Industry and FDA Staff
IEC 60880:2006	Nuclear power plants – instrumentation and control systems important to safety – Software aspects for computer based systems performing category A functions
DO-178B/C ED-12B/C	Software Considerations in Airborne Systems and Equipment Certification

Copyright Notice

Subject to any existing rights of other parties, QA Systems GmbH is the owner of the copyright of this document. No part of this document may be copied, reproduced, stored in a retrieval system, disclosed to a third party or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of QA Systems GmbH.

© QA Systems GmbH 2012

Contents

1 Introduction 5

2 General Information on Cantata 6

2.1 MAJOR FEATURES 6

2.2 FURTHER INFORMATION 6

2.3 CERTIFICATION / QUALIFICATION TERMINOLOGY 7

2.4 TOOL CERTIFICATION 7

2.5 TOOL QUALIFICATION 8

2.5.1 DO-178B Tool Qualification *8*

2.5.2 DO-178C / DO-330 Tool Qualification *8*

2.5.3 Tool Qualification Kit *8*

3 Safety Workflow 9

3.1 STAGES IN SOFTWARE DEVELOPMENT LIFECYCLE 9

3.2 FORMAT OF USE CASES 10

3.3 INSTALLATION AND DEPLOYMENT OF CANTATA 11

3.3.1 Preconditions *11*

3.3.2 Use Case descriptions *11*

3.3.3 Constraints *12*

3.3.4 Dependencies *13*

3.3.5 Post conditions *13*

3.3.6 Consequences of using an unregistered Cantata deployment *13*

3.4 CONTROL OF CANTATA ARTEFACTS 15

3.4.1 Preconditions *16*

3.4.2 Use Case descriptions *16*

3.4.3 Constraints *16*

3.4.4 Dependencies *16*

3.4.5 Post conditions *16*

3.5 USE OF CANTATA OPTIONS 17

3.5.1 Preconditions *17*

3.5.2 Use Case descriptions *17*

3.5.3 Constraints *17*

3.5.4 Dependencies *17*

3.5.5 Post conditions *17*

3.6 USE OF CANTATA STATIC METRICS 18

3.6.1 Preconditions *18*

3.6.2 Use Case descriptions *18*

3.6.3 Constraints *19*

3.6.4 Dependencies *19*

© Copyright QA Systems GmbH 2012

3.6.5 Post conditions 19
3.7 USE OF CANTATA PRAGMAS IN SOURCE CODE 20
3.7.1 Preconditions 20
3.7.2 Use Case descriptions 20
3.7.3 Constraints 20
3.7.4 Dependencies 20
3.7.5 Post conditions 21
3.8 CODE COVERAGE IN CANTATA TESTS 23
3.8.1 Preconditions 23
3.8.2 Use Case descriptions 24
3.8.3 Constraints 26
3.8.4 Dependencies 26
3.8.5 Post conditions 26
3.9 CODE COVERAGE OUTSIDE OF CANTATA TESTS 27
3.9.1 Preconditions 27
3.9.2 Use Case descriptions 28
3.9.3 Constraints 30
3.9.4 Dependencies 30
3.9.5 Post conditions 30
3.10 TESTABILITY INSTRUMENTATION IN CANTATA TESTS 31
3.10.1 Cantata Wrappers and Isolates 31
3.10.2 Test Script Access 32
3.10.3 Preconditions 33
3.10.4 Use case descriptions 34
3.10.5 Constraints 34
3.10.6 Dependencies 34
3.10.7 Post conditions 35
3.11 CREATION OF CANTATA TESTS 35
3.11.1 Preconditions 35
3.11.2 Use case descriptions 35
3.11.3 Constraints 36
3.11.4 Dependencies 36
3.11.5 Post conditions 36
3.12 EXECUTION AND ANALYSIS OF CANTATA TESTS 37
3.12.1 Preconditions 37
3.12.2 Use Case descriptions 37
3.12.3 Constraints 39
3.12.4 Dependencies 39
3.12.5 Post conditions 39

© Copyright QA Systems GmbH 2012

1 Introduction

This Safety Manual provides guidance for how to use the Cantata tool in a safety related context. It should be read in conjunction with the Cantata Manual, and the relevant Cantata Standard Briefing paper.

The Standard Briefing papers below provide a comprehensive mapping of the methods required by that safety standard to the capabilities of the Cantata tool.

Standard Reference	Standard Title	Standard Briefing Paper
IEC 61508:2010	Functional safety of electrical/electronic/programmable electronic safety related systems	Cantata Standard Briefing: IEC 61508-2010
ISO 26262:2011	Road Vehicles – Functional Safety	Cantata Standard Briefing: ISO 26262:2011
EN 50128:2011	Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems	Cantata Standard Briefing: EN 50128:2011
IEC 62304:2006	Medical device software – Software lifecycle processes	Cantata Standard Briefing: IEC 62304:2006
FDA UCM085281:2002	General Principles of Software validation; Final Guidance for Industry and FDA Staff	Cantata Standard Briefing: FDA Software Validation Guidelines
IEC 60880:2006	Nuclear power plants – instrumentation and control systems important to safety – Software aspects for computer based systems performing category A functions	Cantata Standard Briefing: IEC 60880:2006
DO-178B / ED-12B	Software Considerations in Airborne Systems and Equipment Certification	Cantata Standard Briefing: DO-178B / ED-12B

© Copyright QA Systems GmbH 2012

2 General Information on Cantata

Cantata provides advanced high productivity techniques, allowing developers to dynamically and repeatedly prove their C/C++ code with intelligent unit and integration testing, in the most cost effective manner.

The Cantata Eclipse user interface provides a complete intelligent unit and integration testing environment for the creation, execution and analysis of tests. It easily integrates with the developer desk top, compilers and embedded target platforms.

Unit testing and low level integration testing are the earliest tests performed during software development. Intelligent unit and integration testing with Cantata allows developers to fix software bugs at the least expensive time in the development lifecycle, and can be done before bugs pollute the wider code base, or affect shipped devices.

2.1 Major Features

Static Metrics on C & C++ source code

Unit & Integration testing

- Supports C & C++ programming languages
- Testing on development host and embedded targets

Highly automated testing

- Re-usable test harness
- Test script generation written in C/C++
- Unique call interface control to simulate and intercept calls
- Automatic white-box accessibility
- Large data sets and robustness testing

Automated regression testing

Automatic generation of complete baseline of unit tests for legacy C code

The most advanced integrated code coverage analysis available in any tool for C/C++

2.2 Further Information

For current information on Cantata capabilities please refer to the product website www.qa-systems.com/cantata or contact your supplier.

© Copyright QA Systems GmbH 2012

2.3 Certification / Qualification Terminology

Cantata is comprised of both Core Product and Ancillary Tool components which are defined in the Release Notes for each version.

Core Product components are the minimum necessary set required to build and execute dynamic tests, undertake code coverage analysis and generate static code metrics in a safety related context. These components produce evidence (dynamic test reports and comma separated value files of metrics) of the verification activities which are suitable for certification of the users' software.

The Cantata Core Product components may be fully qualified / certified for use against safety related standards.

Ancillary Tools are productivity and diagnostic aids only. The productivity aids (e.g. Eclipse plug-ins to produce template test scripts) may be used in a safety related context provided that the outputs from their use is reviewed. Diagnostic aids (e.g. code coverage displays) must not be used as results for certification evidence of verification activities.

2.4 Tool Certification

Cantata Core Product is currently undergoing comprehensive assessment by an accredited independent third party tool certification body, for usability in development of safety related software against the highest safety integrity levels of the following standards:

Reference	Title	Level	Tool Class
IEC 61508:2010	Functional safety of electrical/electronic/programmable electronic safety related systems	Up to SIL 4	T2
ISO 26262:2011	Road Vehicles – Functional Safety	Up to ASIL D	TCL 1
EN 50128:2011	Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems	Up to SW-SIL 4	T2
IEC 62304:2006	Medical device software – Software lifecycle processes	Up to SW Safety Class C	
FDA UCM085281:2002	General Principles of Software validation; Final Guidance for Industry and FDA Staff		
IEC 60880:2006	Nuclear power plants – instrumentation and control systems important to safety – Software aspects for computer based systems performing category A functions	Category A	Verification and Validation

© Copyright QA Systems GmbH 2012

2.5 Tool Qualification

Tool qualification of Cantata is available where tool certification against a standard is not permitted as the tool must be qualified separately for each project (e.g. for DO-178B / ED-12B), or where a customer wishes to qualify the tool themselves.

2.5.1 DO-178B Tool Qualification

As for all other tools, Cantata may not be certified for general use against the DO-178B / ED-12B standard, but must be Qualified for use on each project. Cantata has been qualified on DO-178B ED-12B projects up to Software Level A. To assist with this the following are papers are available from QA Systems:

- Cantata Standards Briefing: DO-178B / ED-12B

  This serves as a reference to show how Cantata can be used to satisfy the verification and validation requirements of the standard. It maps the requirements of the standard to the capabilities of Cantata, and describes in detail the process for tool qualification of Cantata on a specific project.

  As with all Cantata Standard's Briefings, it should be read in conjunction with the relevant standard, and this Cantata Safety Manual.

- Cantata Tool Qualification Kit – see below

2.5.2 DO-178C / DO-330 Tool Qualification

QA Systems are in the process of updating Cantata tool qualification materials for DO-178C and the new standard DO-330 Software Tool Qualification Considerations. Please contact QA Systems for the latest information regarding these standards.

2.5.3 Tool Qualification Kit

The Cantata tool has been proven in use on a number of safety-related software applications. Customer case studies are available at the product website: www.qa-systems.com/cantata

A tool qualification kit is available free of charge for customers wishing to qualify the tool themselves under a Non-Disclosure Agreement. Please contact QA Systems or your supplier for more details.

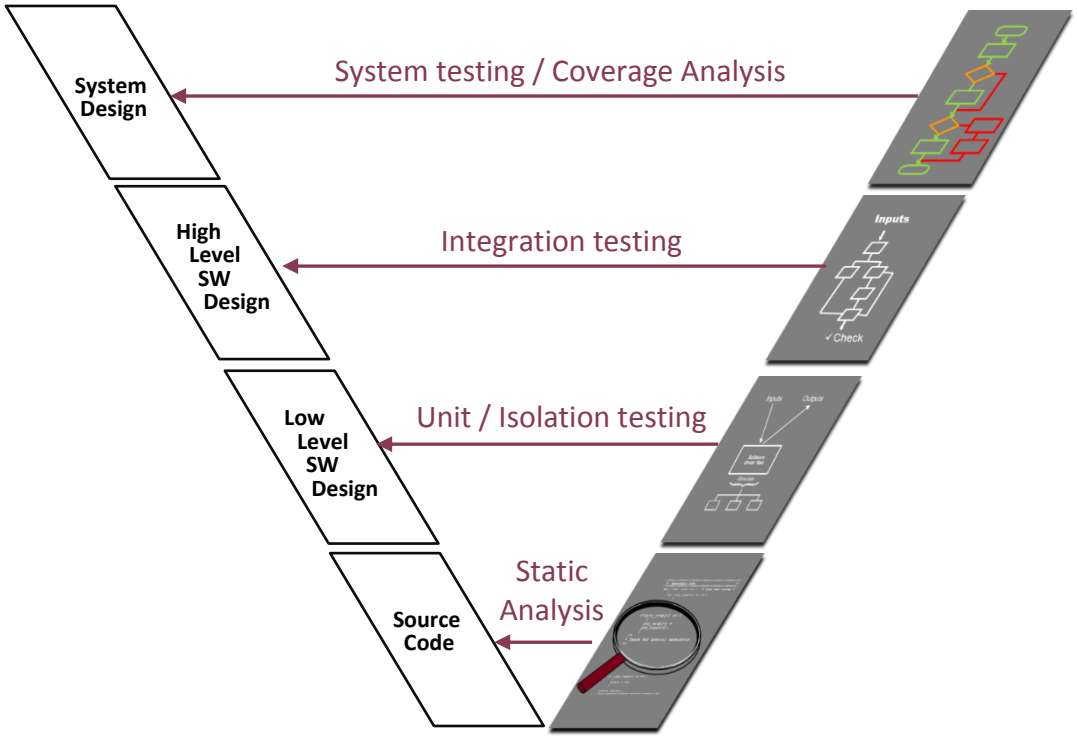
© Copyright QA Systems GmbH 2012

3 Safety Workflow

The section provides guidance on the use of Cantata in a safety related context, by each activity in the workflow. In some cases the Software Level, Software Safety Class or Safety Integrity Level (generically referred to hereafter as SIL), of the software will also be a factor in how Cantata should be used.

3.1 Stages in software development lifecycle

Cantata may be used with any development lifecycle model from traditional waterfall to iterative methods. The V-model figure below indicates the most appropriate workflow activity supported by Cantata at each level of software verification.



An overview Cantata workflow for each software verification activity in a safety related context is described in the following sections.

Verification activity	Relevant Use Cases
Static Analysis	3.3, 3.4, 3.5, 3.7
Unit /Isolation Testing	3.3, 3.4, 3.7, 3.8, 3.10, 3.11, 3.12
Integration Testing	3.3, 3.4, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12
System Testing / Coverage Analysis	3.3, 3.4, 3.7, 3.9, 3.10, 3.12

QA SYSTEMS The Software Quality Company

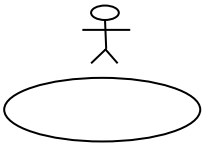
© Copyright QA Systems GmbH 2012

3.2 Format of Use Cases

The Cantata workflow is presented as the following Use Cases:

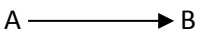
- Installation and deployment of Cantata
- Control of Cantata artefacts
- Use of Cantata options
- Use of Cantata static metrics
- Use of Cantata pragmas
- Code coverage in Cantata tests
- Code coverage outside of Cantata tests
- Testability instrumentation in Cantata tests
- Creation of Cantata tests
- Execution and analysis of Cantata tests

Each Use Case includes preconditions, use case description, constraints, dependencies and post conditions. The Use Case diagrams have the following notation:

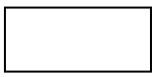


Actor

Use Case [N] – denotes description



B uses results of A



System / Tool

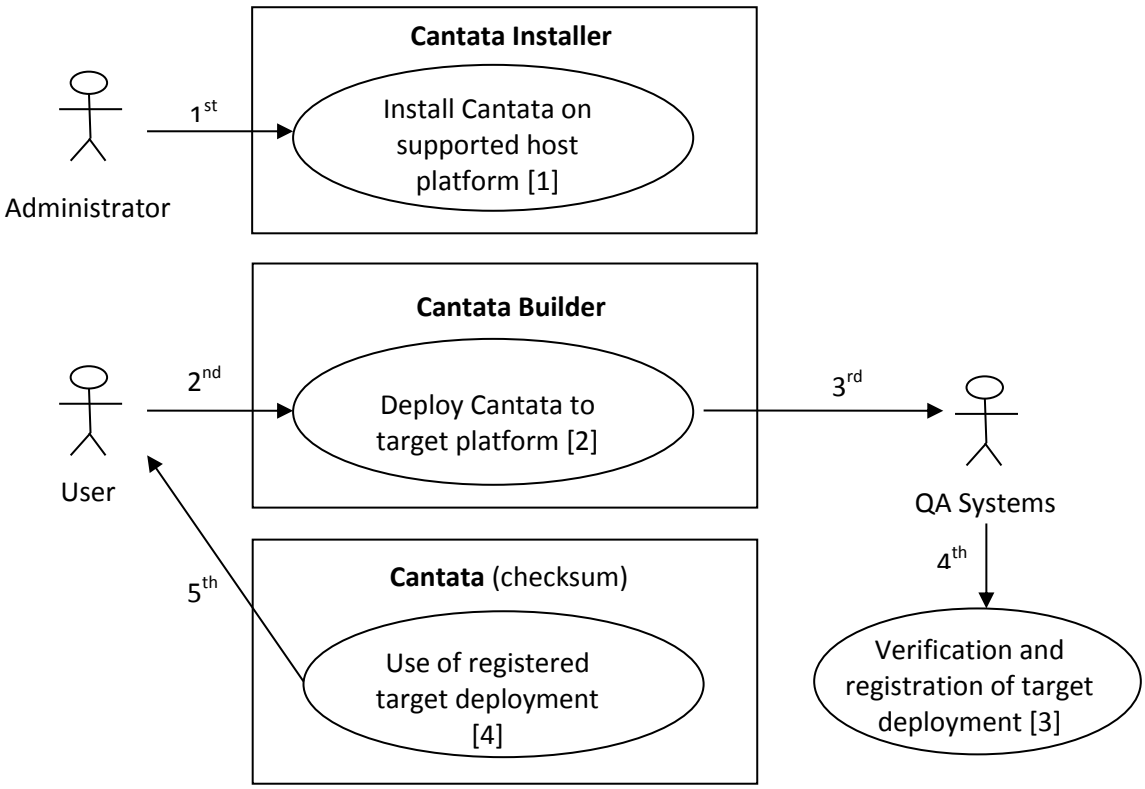
Sequence of use cases

Where the description of a Use Case refers to product documentation for detailed instructions the referenced documentation is underlined.

© Copyright QA Systems GmbH 2012

3.3 Installation and deployment of Cantata

In order to use Cantata for a safety related project it is necessary that the tool is installed on a host platform and deployed correctly for execution of the tests on a target platform.



3.3.1 Preconditions

The version of Cantata can be installed for the following host platforms:

- Operating Systems: Windows (XP, Vista, 7) / Linux (2.4 or 2.6 kernels) / Sparc Solaris (8 - 10)
- Compilers: Microsoft Visual C++ 6.0 – 2010 / GNU GCC g++ (3.x – 4.6)

Cantata can be deployed to a wide range of embedded target platforms either by the user or by a dealer. A Cantata Target Platform Deployment Questionnaire may be completed and supplied to the dealer to determine whether the necessary conditions exist for the successful deployment of Cantata on a target, and to identify in advance any limitations of functionality arising from limitations in the target environment.

3.3.2 Use Case descriptions

[1] Correct installation of Cantata onto a supported host development platform must follow the instructions in the Cantata Quick Start Guide and the Cantata Installation Guide supplied with the product installer.

© Copyright QA Systems GmbH 2012

Correct deployment of Cantata for use on a specific target platform (processor, operating system, compiler and the same options used for building the software.

[2] Correct deployment of Cantata for use on a specific target platform (processor, operating system, compiler and the same options used for building the software under test) must follow the Cantata Deployment Guide section of the Cantata Manual. The deployment to each target platform must be verified with the Cantata Deployment Tests generated for that target platform.

[3] A Cantata Deployment zip file created by the user in use case [2] must be sent to QA Systems for registration.

Verification that the required set of Cantata Deployment Tests have been generated and executed successfully for the target platform is undertaken by QA Systems using the user supplied deployment zip file. Where the results were as expected, the user is sent a registration checksum for the target deployment configuration, and an updated deployment report stating that it has been registered.

[4] Use of a registered deployment is partially enforced by a registration checksum.

Each user's development environment must be configured to select the correct version installation of Cantata and the deployment configuration. This may be verified using any Cantata test on that environment run with the `--verbose` option.

3.3.3 Constraints

The configuration section of the cantpp.cfg file must not contain the option `PORTING=YES` above the checksum (as this would permit use of an unregistered deployment). The deployment may also be safely modified only in the following ways without the checksum failing:

Cantata Option	Modification Safety
`PRODUCT_VERSION`	Should not be changed – provides important information for configuration control
`DEFAULT_COMP_OPTS`	Safe to change but must verify manually that the header files are the same as used for the deployment
`DEFAULT_LIB_EXT`	Safe to change as only used when building library
`INCLUDE_VARIABLE`	Safe to change but must verify manually that the header files are the same as used for the deployment
`LINK_LIBRARY_VARIABLE`	Safe to change but must verify manually that the library files are the same as used for the deployment

© Copyright QA Systems GmbH 2012

Cantata Option	Modification Safety
LIBRARY_DIRECTORY	Safe to change if verified manually
BUILD_PREFIX	
C_FILE_EXTS	
CPP_FILE_EXTS	
OBJ_FILE_EXTS	
REQUIRED_COMPILER_DEFINES	
CCDOBJOPTMAP	
CCDOPTMAP	
CCOPTMAP	
SAOPTMAP	
TOOLOPT	
TOOLOPTMAP	

If any of the above options are modified in other ways, the changed Cantata deployment must be re-verified and registered by QA Systems.

3.3.4 Dependencies

To minimise the need for changes to a deployment configuration in a safety related context it is recommended that the deployment extra parse tests are also run. These parse all the system include files with Cantata. This will identify in advance of testing, any code which cannot be tested with Cantata, and to identify any changes to the Cantata deployment which may be required to accommodate all system include files.

3.3.5 Post conditions

Updates to a deployment's configuration require submission of an updated set of deployment files in a zip file which are re-verified by QA Systems, before a replacement registration checksum is issued to continue using the tool.

3.3.6 Consequences of using an unregistered Cantata deployment

The Cantata deployment registration service is provided to better enable technical support of the product, and assist end users in ensuring that the deployment is correct for their needs (including in a safety related context). QA Systems do not charge for this service, and strongly recommend that all target deployments are therefore registered with QA Systems. If a Cantata deployment is not registered with IPL, then any use of Cantata with that deployment is defined by the click to accept End User Licence Agreement (EULA) as being on an Unauthorised Platform.

© Copyright QA Systems GmbH 2012

EULA Definitions (1.5)

"Authorised Platform" means any Platform either: (i) specified in a Licence File supplied to You by Dealer, or (ii) any Platform which has been verified by successful completion of the deployment tests set out in the Software Product documentation or validation tests supplied to You by Dealer..

EULA 1.20 Definitions (1.20)

“Unauthorised Platform” means any Platform running the Software Product which is not an Authorised Platform.

This has the following consequences defined in the EULA:

EULA 4.3.1 Delivery, Installation and Acceptance (4.3.1)

Dealer agrees to deliver to Your location one copy of each Software Product on an appropriate computer readable medium together with the necessary documentation to use the same and, at Dealer’s option, You agree to allow Dealer to install the Software Product upon the Authorised Platform (if any) specified in the Licence File. Dealer shall have no obligation to install upon an Unauthorised Platform.

EULA Maintenance and Support (4.4.1)

Dealer will provide the Maintenance and Support defined in this clause for any Authorised Platform ONLY during the warranty period and thereafter subject to the payment of the annual Maintenance and Support charges.

EULA Maintenance and Support (4.4.6)

Maintenance and Support shall be provided for the Software Product on any Authorised Platform during the Rental period as long as the Rental charges are paid.

EULA Maintenance and Support (4.4.8)

Notwithstanding anything else in this Agreement and/or payment by You of any Maintenance and Support charges, Dealer shall have no obligation to provide Maintenance and Support for any Platform, and any Platform upon which You are running the Software Product shall be deemed an Unauthorised Platform, unless You can provide promptly upon Dealer’s request in respect of that Platform evidence of either: (i) a valid licence File or (ii) successful verification of the Platform by providing the report produced by the Software Product upon completion of the deployment tests set out in the Software Product documentation or validation tests supplied to you by Dealer.

EULA Maintenance and Support (4.5.2)

Dealer warrants that the Software Product will be substantially capable of performing the function described in the user documentation upon Authorised Platform(s) only.

EULA Maintenance and Support (4.5.4)

Dealer warrants that it will use commercially reasonable endeavours to correct errors in the Software Product running on Authorised Platform(s) free of charge during the warranty period and any subsequent periods for which annual Maintenance and Support charges have been paid.

EULA Maintenance and Support (4.5.5)

All warranties will be null and void if: (a) You have modified the Software Product in any way, or (b) You are running the Software Product on an Unauthorised Platform.

In addition QA Systems shall not provide any assistance to a user in support of any tool certification/qualification exercise involving Cantata deployments not registered with QA Systems.

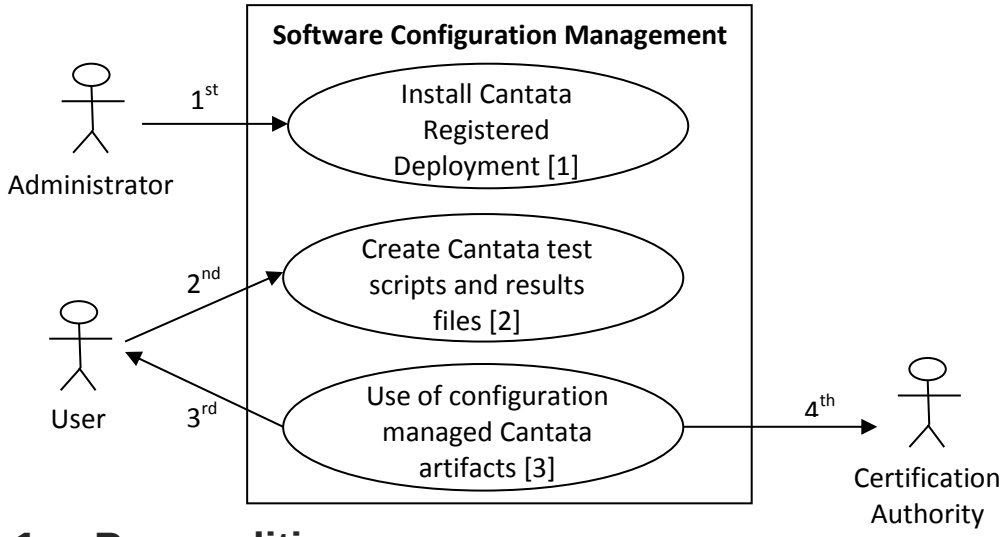
© Copyright QA Systems GmbH 2012

3.4 Control of Cantata artefacts

How Cantata artefacts are controlled will form part of the certification evidence that the tool has been used correctly in a safety related environment. Below are listed the Cantata artefacts which should be configuration controlled:

Artefact	Description
Cantata build.txt installation file	Records 4 digit build number in the root installation directory
Registered tool configuration file sections (cantpp.cfg) and deployment header file `<configuration_section_name>.h`	Defines the platform(s) on which Cantata was configured and on which tests were run.
Registered Cantata Deployed Library	Library built for the platform on which the Cantata Deployment Tests and Cantata tests were run.
Registered Cantata Deployment Report	Defines the deployment and Cantata Deployment Tests for the platform(s), and any limitations of Cantata on the platform(s) arising from limitations of the platform(s).
Cantata Test Scripts	Test Scripts and any associated Cantata header files which were executed.
Cantata ASCII test results files `<test_name>.ctr`	Cantata test results produced by the Core Product components direct from target, and suitable for certification evidence purposes.
CSV files of metrics	Metrics on source code produced by Cantata static analysis.
Cantata Makefiles	Cantata Makefiles (where they are used), control the building and invocation of test executables. Where Cantata Makefiles are not used, the alternative method for building and invoking test executables should be configuration controlled.
Test project options files (`ipg.cop`)	All Cantata test project options files.
Target Deployment Technical Note	Any target specific instructions supplied for building and executing Cantata tests on the registered target deployment.

© Copyright QA Systems GmbH 2012



3.4.1 Preconditions

Above listed artefacts are created.

3.4.2 Use Case descriptions

[1] Correct installation of a Cantata Registered Deployment to the Configuration File (cantpp.cfg) must follow the Cantata Deployment Guide section of the Cantata Manual.

[2] Cantata Test Scripts may be created in any code editor or using the Cantata user interface as described in section 3.11 and defined in the Cantata Manual.

Cantata results files are generated by executing a Cantata test as described in section 3.11 and defined in the Cantata Manual.

[3] Cantata artefacts consist of inputs to tests and outputs from them. A configuration control system must be used to version control all Cantata artefacts for testing.

The managed Cantata artefacts must be provided to a certification authority as evidence of the verification activities undertaken using Cantata.

3.4.3 Constraints

Cantata artefacts are time stamped. It is recommended that Cantata Test Scripts, associated header files and ipg.cop file(s) are version controlled at the same version as the source code under test to which they relate.

3.4.4 Dependencies

None

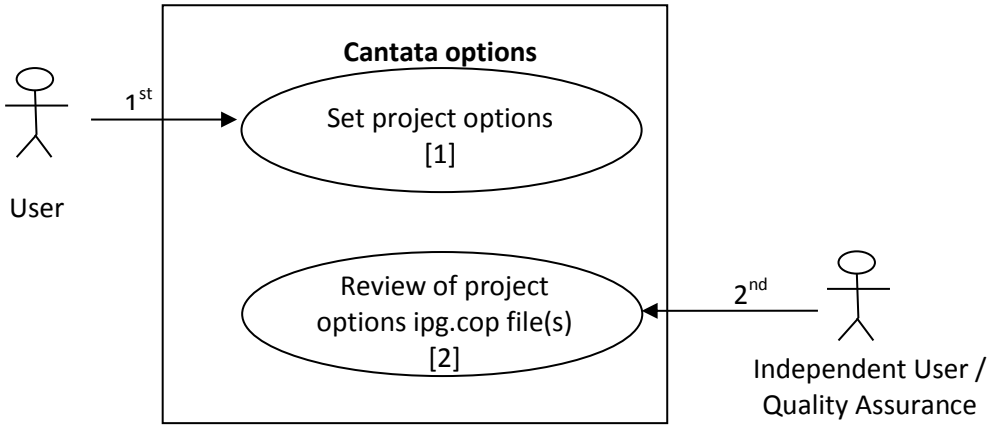
3.4.5 Post conditions

Configuration records for each Cantata artefact associated with the tested source code.

© Copyright QA Systems GmbH 2012

3.5 Use of Cantata options

The set of Cantata project options used when building and executing Cantata tests are stored in ipg.cop file(s). The configuration section for the specific target deployment used in the test is recorded within the ipg.cop file(s). This review should ensure that these Cantata options are applicable for the appropriate SIL for the safety related project. All Cantata options are defined in the Cantata Manual.



3.5.1 Preconditions

The Cantata project options ipg.cop file(s) used for a test are configuration controlled together with the test script, any associated header files and source code under test.

3.5.2 Use Case descriptions

[1] Cantata project options may be set using: Eclipse Workspace Preferences, as specific Project Properties as described in the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the ipg.cop file(s) directly.

[2] The Cantata project options (including the referenced deployment configuration section of the cantpp.cfg file) must be independently reviewed as inputs to the test according to the option definition in the Cantata Manual.

3.5.3 Constraints

The ipg.cop file contains a warning not to modify the file directly, however if this is done, then only the section marked `#User Section` may be safely edited.

3.5.4 Dependencies

The Cantata log level (which defaults to normal) must be set to minimal or higher (i.e. not to None), as this will ensure that the any `CHECK` directive resulting in a fail will be reported in a result file `<test_name>.ctr` for the test.

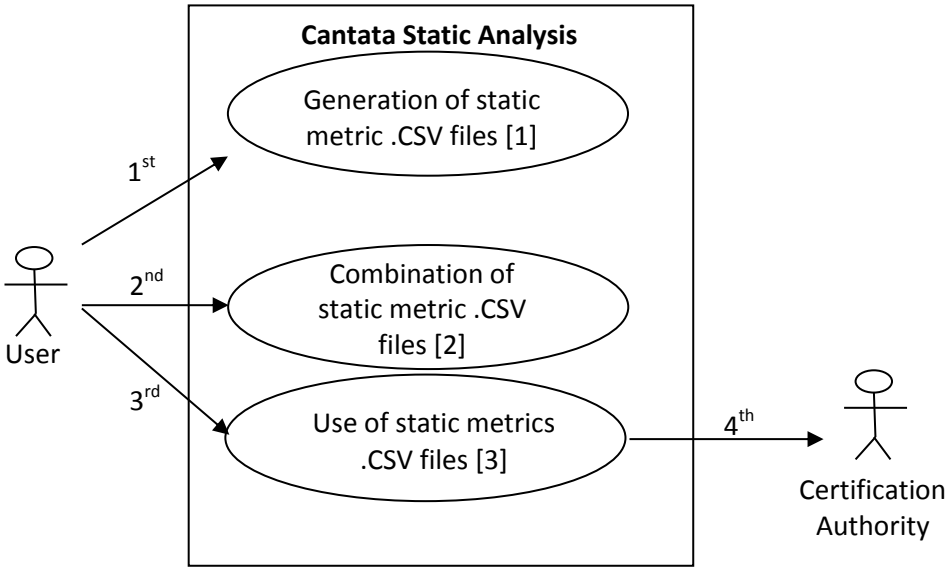
3.5.5 Post conditions

Documented review of Cantata project options ipg.cop file(s).

© Copyright QA Systems GmbH 2012

3.6 Use of Cantata static metrics

The Cantata Static Analysis tool may be used to generate metrics on C/C++ source code.



3.6.1 Preconditions

An installation of Cantata must exist. Source code must exist in a Cantata enabled project.

3.6.2 Use Case descriptions

[1] Cantata static metrics are generated as a single comma separated value .CSV file for each source file by building the source code in a Cantata enabled project with static analysis options in the ipg.cop project options file.

These options may be set in the user interface as defined in the Project Properties within the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the ipg.cop file(s) directly.

[2] Where the static analysis is applied to multiple files, individually generated .CSV files must be combined using the CSV Analysis Tool [analyse.exe (windows) / analyse.awk (Linux)], as defined in the Static Metrics - CSV Analysis section of the Cantata Manual. The combined .CSV file contains all the information from the individual input .CSV files along with additional static analysis data generated through the analysis of the interactions between the objects in the software under test.

[3] Where static metrics are used for analysis of the source code, a configuration control system must be used to version control the .CSV files. The .CSV files must be provided to a certification authority as evidence of the analysis activities undertaken using Cantata.

© Copyright QA Systems GmbH 2012

3.6.3 Constraints

The following 9 out of 300+ Cantata static analysis metrics do not work correctly and therefore cannot be used on a safety related project:

Metric Name	Description
STMT_NULL	Number of null statements
STATEMENTS	Total number of statements
NESTING_AVERAGE	Average statement nesting levels over all statements in the function
UNREACHABLE	Number of unreachable statements
OPER_OTHER	Number of other operator uses (all operators other than ?:, &&, ||, ! and overloaded operators)
UNIQREF_EXTCONST	Number of non-local non-member constants referenced
UNIQREF_OWNCONST	Number of member constants defined in the class of which this function is a member which are referenced
TOTALREF_EXTCONST	Number of non-local non-member constants references
TOTALREF_OWNCONST	Number of references to member constants defined in the class of which this function is a member

The following Ancillary Tools are not part of the certified Cantata Core Product and are therefore not suitable to use for safety related projects.

Tool	Description
csvmerge.exe	Used to combine individual .csv files
Cantata Static Analysis Presentation Tool	Excel macros used to generate graphs from source .csv files

3.6.4 Dependencies

None.

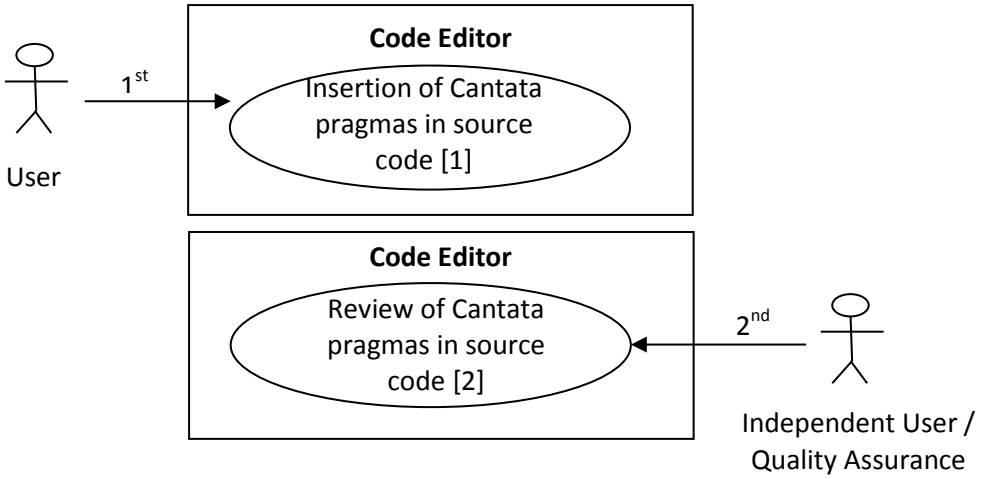
3.6.5 Post conditions

.CSV files for each analysed source file containing metric name and counts for the selected static metrics.

© Copyright QA Systems GmbH 2012

3.7 Use of Cantata pragmas in source code

Cantata pragmas may be used to modify the way Cantata treats parts of the source code.



3.7.1 Preconditions

The use of Cantata pragmas in source code is independently reviewed and justified in the context of the associated Cantata test and SIL for the safety related context.

3.7.2 Use Case descriptions

[1] Cantata pragmas must be inserted manually into the source code, as defined in Appendix H of the Cantata Manual and must only use the code comment form of the pragmas (e.g. `/* #pragma ipl Cantata ignore on */`)

[2] Cantata Pragmas inserted in source code must be independently reviewed as inputs to the test.

3.7.3 Constraints

If source code is never to be modified for testing in the safety related context, then Cantata pragmas must not be used.

3.7.4 Dependencies

None.

© Copyright QA Systems GmbH 2012

3.7.5 Post conditions

The Cantata pragmas used around a section of source code have the following purpose and effects:

3.7.5.1 Parsing control pragmas

```
/* #pragma ipl Cantata ignore on */
/* #pragma ipl Cantata ignore off */
/* #pragma ipl Cantata predefine on */
/* #pragma ipl Cantata predefine off */
```

Purpose	Effect
The ignore pragma pair permits parsing of non-standard source code constructs	Silently excludes code from Cantata parsing and all analysis
The predefine pragma pair allows definitions to be passed to the parser but not to the compiler, preventing redefinition errors.	Silently treats defines as if they were predefined macros.

3.7.5.2 Instrumentation pragmas

```
/* #pragma ipl Cantata ci instr <metric type> on */
/* #pragma ipl Cantata ci instr <metric type> off */
/* #pragma ipl Cantata ci instr <metric type> default*/
```

Purpose	Effect
To switch on/off or coverage instrumentation by metric type	Silently excludes code from instrumentation for coverage analysis

3.7.5.3 Infeasible pragmas

```
/* #pragma ipl Cantata ci infeasible <metric> on */
/* #pragma ipl Cantata ci infeasible <metric> off */
/* #pragma ipl Cantata ci infeasible <metric> on in context "name" */
/* #pragma ipl Cantata ci infeasible <metric> off in context "name" */
```

To report coverage information on source code marked as infeasible using the above pragmas, the Cantata `REPORT_COVERAGE` coverage directive option `cppca_include_infeasible` must be used:

© Copyright QA Systems GmbH 2012

Purpose	Effect

Controls coverage reporting applied to source code within the pragma, which is not expected to be exercised by the test, or within a certain named context.
 	
Coverage reports can be generated on this basis, making "100% feasible coverage" a meaningful and enforceable test requirement, where the Cantata `GET_COVERAGE` coverage directive excludes the `cppca_include_infeasible` option.
	Coverage results report coverage as `including infeasible`
	
	Each coverage percentage which is affected by an infeasible case is marked with a *.
	Before each line containing an affected percentage the following is reported:
	`>> WARNING: N infeasible <metric>`
	
	Each element within the code marked as infeasible by the pragma is reported as:
	`>> INFEASIBLE`
	
	The count of code elements is recorded as
	`>> EXECUTED INFEASIBLES`

3.7.5.4 Other pragmas

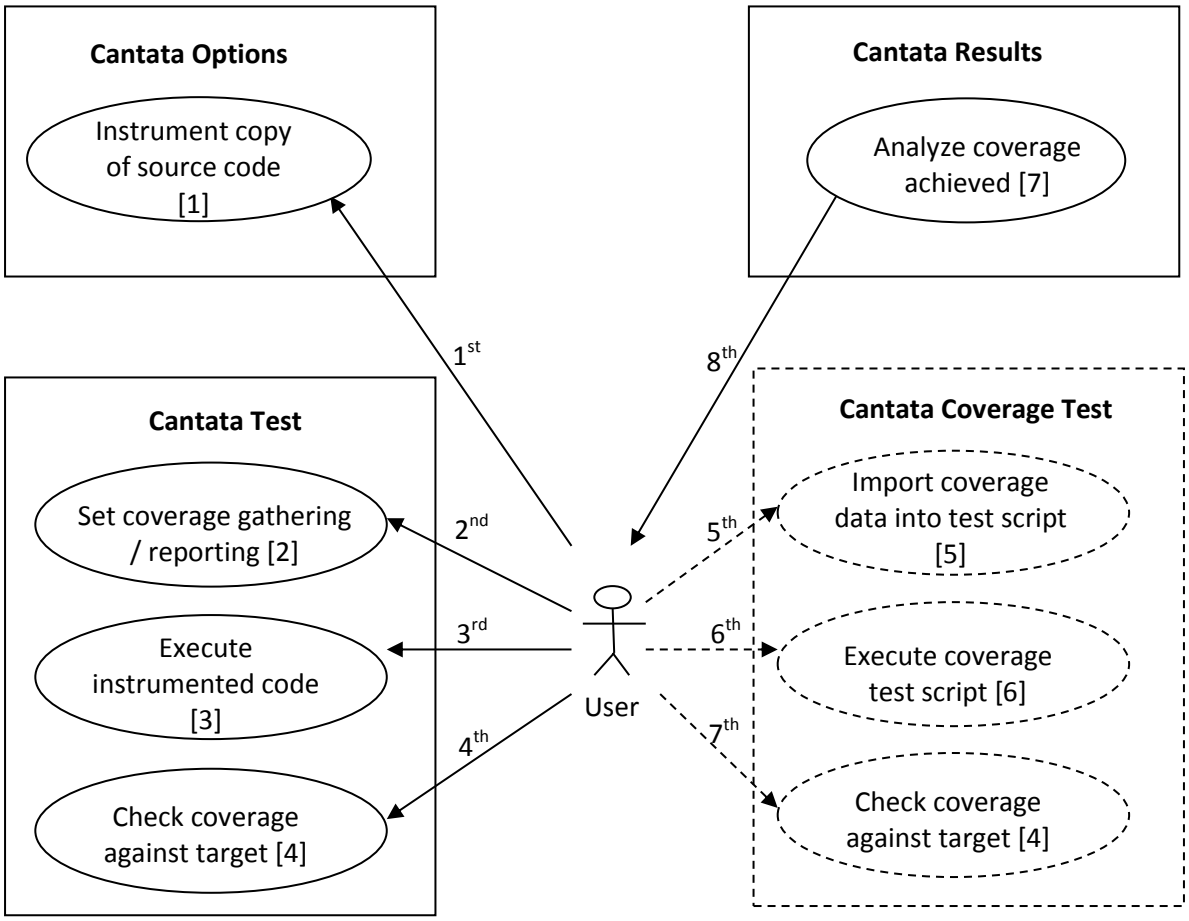
```
/* #pragma ipl Cantata sm wrap on */
/* #pragma ipl Cantata sm wrap on */
/* #pragma ipl Cantata sm wrapable on */
/* #pragma ipl Cantata sm wrapable off */
/* #pragma ipl Cantata sm friend on */
/* #pragma ipl Cantata sm friend off */
/* #pragma ipl Cantata sm clone on */
/* #pragma ipl Cantata sm clone off */
/* #pragma ipl Cantata sm wrap default */
/* #pragma ipl Cantata sm wrappable default */
/* #pragma ipl Cantata sm friend default */
/* #pragma ipl Cantata sm clone default */
```

Purpose	Effect
Controls wrapping of specific function calls applied to section of source code	Overrides wrapping options specified in project options ipg.cop file(s)
Controls friend declaration applied to class definitions within the pragma	Overrides friend options specified in project options ipg.cop file(s)
Controls clone modifications applied to class definitions within the pragma	Overrides clone modification options specified in project options ipg.cop file(s)
Restores settings selected by wrap friend or clone options	Restores the settings for wrap friend or clone options to those in the project options ipg.cop file(s)

© Copyright QA Systems GmbH 2012

3.8 Code coverage in Cantata tests

Code coverage analysis in Cantata tests is used to measure the code constructs and proportions within the source code, resulting from execution of the Cantata test.



Where coverage gathering and reporting is not supported on the target platform

Or

Coverage analysis is combined over multiple Cantata test scripts.

3.8.1 Preconditions

A Cantata registered deployment for target platform must be available.

Where a Cantata deployment does not support coverage checking on the target platform, a Cantata registered deployment for the host operating system with a host compiler must be available to execute a Cantata coverage only test.

The Cantata Log level must be set to normal or higher for recording of full code coverage metric details and checks against targets in the `<test_name>.CTR` files.

© Copyright QA Systems GmbH 2012

3.8.2 Use Case descriptions

[1] Cantata source code coverage instrumentation options must be set in the Cantata options file (ipg.cop) for each test project, according to the coverage metrics (code construct) required by the appropriate SIL or quality plans for the safety related project as below.

Minimum Coverage Required	Cantata Coverage Metrics to be used
Statement coverage	Entry-Point + Statement
Decision coverage	Entry-Point + Statement + Decision + Boolean Operator
MC/DC (modified condition / decision coverage)	Entry-Point + Statement + Decision + Boolean Operand Effectiveness (Masking or Unique Cause variant)

Coverage instrumentation options may be set using Code Coverage RuleSets via the user interface as defined in the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the ipg.cop file(s) directly.

[2] Cantata coverage gathering and reporting must be set using Cantata Coverage Gathering and Coverage Reporting directives in the test script as defined in the Coverage Analysis Library Reference section of the Cantata Manual. These directives may be set in Code Coverage RuleSets via the user interface as defined in the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the test script directly.

Where exception handlers are present in the source code, Cantata coverage reporting test script directives must include the `cppca_include_catch` flag for statement coverage, as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

Due to the potential combination of instrumentation options and pragmas, it is possible that all unexecuted functions may not be reported as such. Therefore to ensure that all functions required to be tested have associated code coverage results, one of the approaches below must be adopted:

- When instrumenting for code coverage, the `--ci:list_instrumentation[:filename]` option is used to write to the specified file a list of instrumentation types for each function, as defined in the Coverage Instrumentation Tool Reference section of the Cantata Manual. This provides a full record of the instrumentation undertaken. This list must then be compared to the Cantata reports of code coverage, to ensure there are no gaps.

- When reporting on code coverage an explicit function pattern list (i.e. not *) is used as defined in the Coverage Reporting Library Reference section of the Cantata Manual. This ensures that any un-executed or un-instrumented functions are reported.

© Copyright QA Systems GmbH 2012

Cantata coverage data gathering automatically guards against the possibility of erroneous data corrupting results as defined in the Coverage Gathering Library Reference section of the Cantata Manual.

Cantata coverage data reporting automatically checks that the coverage data is valid to report on for the instrumented source code files used in the test as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

Cantata coverage reporting directives will only report 100% if all coverage items were executed, with the percentage truncated never rounded up,

By default, Cantata coverage items marked as "infeasible" or as "catch-clauses" are not considered when calculating the coverage achieved; optional flags can be specified to force their inclusion in reports and percentage calculations, as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

[3] Instrumented source code must be used in the build and running of the test executable as defined in the Coverage Analysis section of the Cantata Manual.

The same Cantata tests must be executed both with and without coverage instrumentation options enabled, to demonstrate that the instrumentation has not affected the result.

[4] Coverage checking must use Cantata `ANALYSIS_CHECK` macros or the `GET_COVERAGE` directive directly, within the test script for the given coverage type against the given target percentage lower and upper values, both as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

The `ANAYSIS_CHECK` macros may be specified in Code Coverage RuleSets via the user interface as defined in the Cantata User Interface Guide section of the Cantata Manual. The `ANALYSIS_CHECK` macros (which can only used within a `rule_set` function in a test script), and `GET_COVERAGE` directive may also be set by manually editing the test script directly.

[5] Where coverage data is obtained using a Cantata deployment which does not support coverage checking on the target platform, then the coverage file `<test_name>.cov` generated during test execution on the target platform must be processed on the host platform with a separate Cantata coverage test script.

Where coverage data obtained in more than one test script is to be combined, then the coverage file `<test_name>.cov` generated during test execution of each test to be combined must be processed on the host platform with a separate Cantata coverage test script.

In both cases the `IMPORT_COVERAGE`, `GET_COVERAGE` and `REPORT_COVERAGE` Cantata coverage directives must be used in this Cantata coverage test script

© Copyright QA Systems GmbH 2012

If the achieved coverage is to be checked against a target, then the Cantata `GET_COVERAGE_MIN` directive must be used, so that a `CHECK` directive will verify that the minimum coverage within each function has been achieved for each coverage metric type.

[6] A separate Cantata coverage test script must be executed on the registered host platform deployment.

[7] The Cantata Eclipse user interface Ancillary Tools, may be used as detailed diagnostic aids, as defined in the Cantata User Interface Guide section of the Cantata Manual.

The certification evidence for code coverage produced by the Cantata Core Product components directly from the target is only contained in the .CTR test results file which presents the coverage analysis in a structured ASCII file. Analysis of the .CTR file must include achievement of all coverage targets, and absence of both unexecuted code (where not marked infeasible) which are indicated by

`>> NOT EXECUTED`, and `>> Script Error`.

Cantata prevents analysis of coverage results which have a different file 'signature' (location and timestamp) to the source code. Where data is incompatible, diagnostic data not loaded in the Cantata Eclipse user interface, and a `Script Error` is included in the .CTR file.

3.8.3 Constraints

Cantata Boolean Operand Effectiveness coverage analysis only supports logical operators not bit-wise operators.

A Cantata deployment for the host operating system with a host compiler must be used to execute the Cantata coverage test as the Ancillary Tool `cppgetcov` is not part of the Cantata Core Product, so must not be used in a safety related context.

3.8.4 Dependencies

Source code is required for Cantata coverage instrumentation.

3.8.5 Post conditions

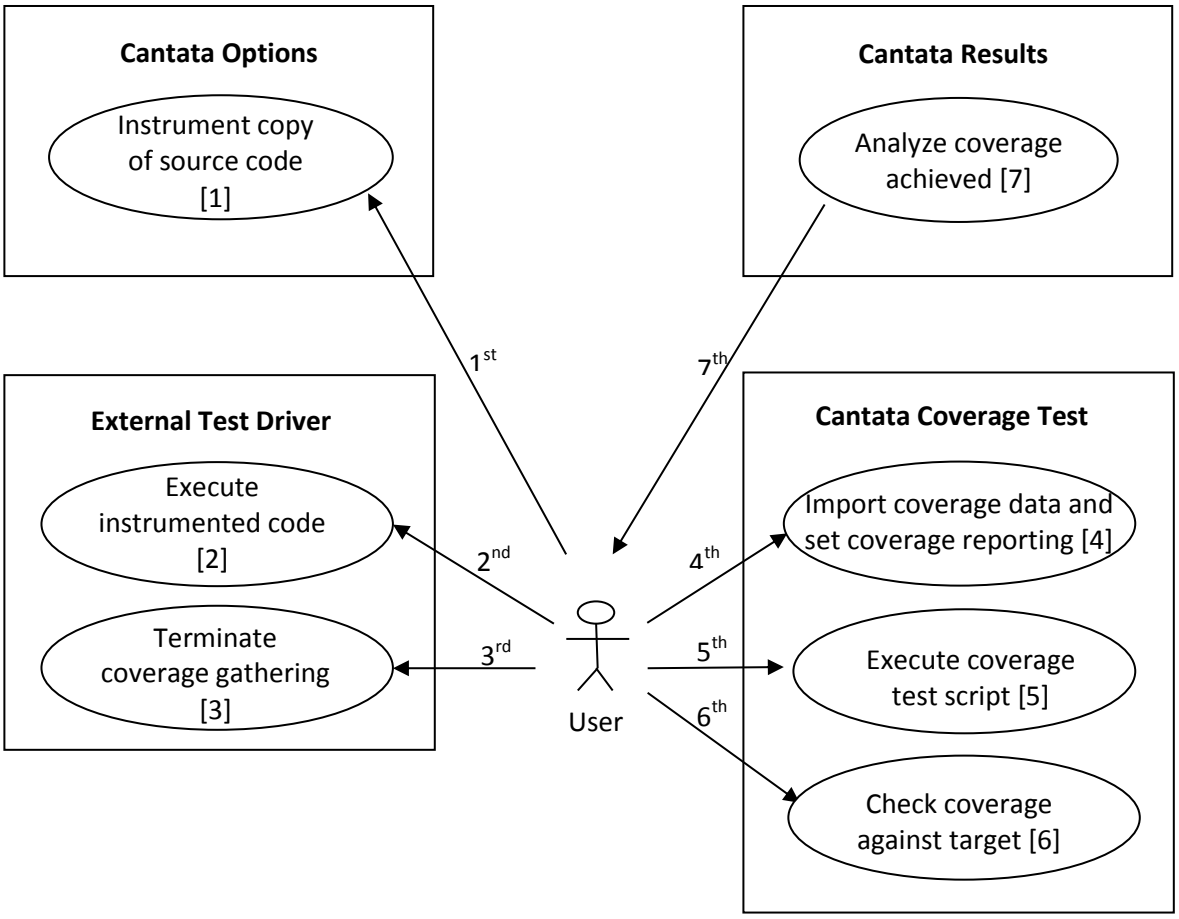
Cantata Ancillary Tools provide for diagnostics of coverage achieved.

In a safety related context only the Cantata ASCII text `<test_name>.CTR` results files produced by the Core Product direct from the target recording the degree of structural coverage (as well as test checks against coverage targets) from Cantata tests run with and without coverage instrumentation must be used as certification evidence.

© Copyright QA Systems GmbH 2012

3.9 Code coverage outside of Cantata tests

Cantata Code coverage analysis may be used to measure the code constructs and proportions within the source code, resulting from execution of the software when driven by something other than a Cantata test.



3.9.1 Preconditions

A Cantata registered deployment for the target and host platforms must be available.

The Cantata Log level must be set to normal or higher (using an environment variable), for recording of full code coverage metric details and checks against targets `<test_name>.CTR` files.

© Copyright QA Systems GmbH 2012

3.9.2 Use Case descriptions

[1] Cantata source code coverage instrumentation options must be set in the Cantata options file (ipg.cop) for the project, according to the coverage metrics (code construct) required by the appropriate SIL or quality plans for the safety related project as below.

Minimum Coverage Required	Cantata Coverage Metrics to be used
Statement coverage	Entry-Point + Statement
Decision coverage	Entry-Point + Statement + Decision + Boolean Operator
MC/DC (modified condition / decision coverage)	Entry-Point + Statement + Decision + Boolean Operand Effectiveness (Masking or Unique Cause variant)

Coverage instrumentation options may be set using the Cantata Eclipse Project Properties via the user interface as defined in the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the ipg.cop file(s) directly.

[2] Instrumented source code must be used in the build and running of the non-Cantata driven executable as defined in the Coverage Analysis section of the Cantata Manual.

The same non-Cantata driven tests must be executed both with and without coverage instrumentation options enabled, to demonstrate that the instrumentation has not affected the result.

[3] A mechanism for terminating coverage gathering for the software under test and thereby generating a Cantata coverage file (default name: `cantpp.cov`), must exist at end of the test run, when not using a Cantata test to execute the software. Cantata coverage gathering will automatically invoke the `EXPORT_COVERAGE` directive on normal program termination (e.g. `exit`, `atexit`) if no other coverage analysis directives (except `CONFIGURE_COVERAGE`) have been used.

Where the non-Cantata driven test does not result in the termination of the program, the `EXPORT_COVERAGE` directive must be invoked by some other means. An example technique is to invoke the `EXPORT_COVERAGE` directive from within a Cantata non-test script wrapper on a call made after the test has been completed.

The code coverage file created by each non-Cantata driven test will be named `cantpp.cov`, this file must be suitably renamed to `<test_name>.cov` after each test is completed otherwise it will be overwritten.

[4] Cantata `IMPORT_COVERAGE`, `GET_COVERAGE` and `REPORT_COVERAGE` coverage gathering and reporting must be set using Cantata Coverage Gathering and Coverage Reporting directives in the test script as defined in the Coverage Analysis Library Reference section of the Cantata Manual. These directives may be set in Code Coverage RuleSets via the

© Copyright QA Systems GmbH 2012

user interface as defined in the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the test script directly.

Where exception handlers are present in the source code, Cantata coverage reporting test script directives must include the `cppca_include_`catch flag for statement coverage, as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

Due to the potential combination of instrumentation options and pragmas, it is possible that all unexecuted functions may not be reported as such. Therefore to ensure that all functions required to be tested have associated code coverage results, one of the approaches below must be adopted:

- When instrumenting for code coverage, the `--ci:list_instrumentation[:filename]` option is used to write to the specified file a list of instrumentation types for each function, as defined in the Coverage Instrumentation Tool Reference section of the Cantata Manual. This provides a full record of the instrumentation undertaken. This list must then be compared to the Cantata reports of code coverage, to ensure there are no gaps.

- When reporting on code coverage an explicit function pattern list (i.e. not *) is used as defined in the Coverage Reporting Library Reference section of the Cantata Manual. This ensures that any un-executed or un-instrumented functions are reported.

Cantata coverage data gathering automatically guards against the possibility of erroneous data corrupting results as defined in the Coverage Gathering Library Reference section of the Cantata Manual.

Cantata coverage data reporting automatically checks that the coverage data is valid to report on for the instrumented source code files used in the test as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

Cantata coverage reporting directives will only report 100% if all coverage items were executed, with the percentage truncated never rounded up,

By default, Cantata coverage items marked as "infeasible" or as "catch-clauses" are not considered when calculating the coverage achieved; optional flags can be specified to force their inclusion in reports and percentage calculations, as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

[5] The Cantata coverage test script must be executed on the registered host platform deployment.

[6] Coverage checking must use Cantata `ANALYSIS_CHECK` macros or the `GET_COVERAGE` directive directly, within the test script for the given coverage type against the given target percentage lower and upper values, both as defined in the Coverage Reporting Library Reference section of the Cantata Manual.

© Copyright QA Systems GmbH 2012

The ANALYSIS_CHECK macros may be specified in Code Coverage RuleSets via the user interface as defined in the Cantata User Interface Guide section of the Cantata Manual. The ANALYSIS_CHECK macros (which can only used within a rule_set function in a test script), and GET_COVERAGE directive may also be set by manually editing the test script directly.

If the achieved coverage is to be checked against a target, then the Cantata GET_COVERAGE_MIN directive must be used (this is contained within the ANALYSIS_CHECK macros), so that a CHECK directive will verify that the minimum coverage within each function has been achieved for each coverage metric type.

[7] The Cantata Eclipse user interface Ancillary Tools, may be used as detailed diagnostic aids, as defined in the Cantata User Interface Guide section of the Cantata Manual.

The certification evidence for code coverage produced by the Cantata Core Product components directly from the target is only contained in the .CTR test results file which presents the coverage analysis in a structured ASCII file. Analysis of the .CTR file must include achievement of all coverage targets, and absence of both unexecuted code (where not marked infeasible) which are indicated by >> NOT EXECUTED, and >> Script error.

Cantata prevents analysis of coverage results which have a different file 'signature' (location and timestamp) to the source code. Where data is incompatible, diagnostic data not loaded in the Cantata Eclipse user interface, and a Script error is included in the .CTR file.

3.9.3 Constraints

Cantata Boolean Operand Effectiveness coverage analysis only supports logical operators not bit-wise operators.

A Cantata deployment for the host operating system with a host compiler must be used to execute the Cantata coverage test, as the Ancillary Tool cppgetcov is not part of the Cantata Core Product, so must not be used in a safety related context.

3.9.4 Dependencies

Source code is required for Cantata coverage instrumentation.

A Cantata test script is required to import, report coverage achieved for the test run.

3.9.5 Post conditions

Cantata Ancillary Tools provide for diagnostics of coverage achieved.

In a safety related context only the Cantata ASCII text <test_name>.CTR results files produced by the Core Product direct from the target recording the degree of structural coverage (as well as

© Copyright QA Systems GmbH 2012

test checks against coverage targets) from tests run with and without coverage instrumentation must be used as certification evidence.

3.10 Testability instrumentation in Cantata tests

Cantata testability instrumentation is used to enable white-box access to code internals for testing.

Cantata provides a Source Modification mechanism run on a copy of the source code under test at build time, which may be used to provide controlled access for the Cantata test script to source code statements and structures which an external test file would not otherwise have. This mechanism increases and automates the ability to verify the software under test, but does require testability instrumentation of the source code. The following verification activities are enabled through Cantata Source Modification, each of which is enabled and disabled by tool options:

- Wrapping and Isolating function calls
- Access for test script to source code data and functions limited to the file boundary (private data, unnamed namespaces and calling private methods in C++, and static data and calling static functions)

Full details on use of Cantata Testability Instrumentation are contained in the Testability Instrumentation section of the Cantata User Manual.

In the sections below, the white-box testing challenges are explained in more detail as they relate to the two verification activities possible using Cantata testability instrumentation through controlled Source Modification.

In a safety related context it is recommended that Cantata testability instrumentation is assessed against the requirements of the SIL, and is explicitly defined and justified where used in the quality plans.

3.10.1 Cantata Wrappers and Isolates

Cantata provides three mechanisms for controlling call interfaces. Stubs and Isolates are simulations of called objects which are used to replace the called object during a test. Wrappers are interceptions of calls to objects, which are used to check or modify what is passed over the call. Cantata Stubs, Isolates and Wrappers are contained in the Cantata test script, and their use of specific programmable instances within them is controlled in each individual test case by the call sequence verification directive `EXPECTED_CALLS`.

Cantata stubs can be used without any form of instrumentation, as the stub code replaces the called object completely. Cantata isolates replace the called object even though it may be linked into the built object. Cantata wrappers require the called object to be linked in the build of the source code.

Both isolates and wrappers require instrumentation of the source code using the Cantata Source Modification tool. This inserts code into a temporary copy of the source code `<file_name.sm.cxx>` for testing with Cantata. It is automated through tool options and does not require conditional compilation of the source code just for testing. This simulation or

© Copyright QA Systems GmbH 2012

interception code then calls the isolate or wrapper code in the test script (either immediately before the object is then called, or to replace it like a stub).

The competing test challenges in a safety related context for the use of Cantata testability instrumentation for wrapping or isolating are:

- That source code may have statements or structures which cannot be verified without simulating or intercepting the return from the called object, which may not be possible or safe with stubs. Examples where calls cannot technically be replaced by stubs, or to attempt to do so imposes too great a risk that the stub simulation of something very complex is incorrect, are:
  - Calls to operating system functions
  - Calls to functions/methods internal to the compilation unit/file boundary
  - Calls to third party software

  In these cases Cantata wrapping or isolating allows the tester to ensure that the source code can be verified, where stub simulation is not even possible or is riskier than using wrapping and using the real called object.

- That it may not be desired for source code to be altered in order to test it, implying that to alter it using Cantata testability instrumentation for wrapping or isolating calls to achieve greater testability may not be acceptable. A distinction may however be made between two purposes specifically of wrapping instrumentation enabled in tests:
  - Wrapping used only to verify parameters passed over interfaces and the order of calls made
  - Wrapping used to modify parameter values or insert other code passed over interfaces in order to force behaviour in the source code

  In the first scenario the same test case can be run twice, with wrapping instrumentation enabled / disabled, provided that separate parallel interleaved expected call sequences are used for stubs and wrappers. If the test results are the same other than the additional checks in wrappers not being executed and wrapping only call sequence validation failures, then the testability instrumentation can be demonstrated not to have affected the execution behaviour of the source code (as for code coverage instrumentation).

  In the second scenario, wrapping instrumentation may only ever be enabled as it will always be used to execute the tests.

3.10.2 Test Script Access

Cantata provides a mechanism for granting the test script direct access to data accessed and calls made by the source code which would otherwise be prevented. It requires instrumentation of the source code using the Cantata Source Modification tool. This instrumentation inserts accessor functions or C++ Friends declarations into a temporary copy of the source code only for testing with Cantata. It is automated through tool options and does not require conditional compilation of the source code just for testing.

The competing test challenges in a safety related context for use of code accessibility for the test script are:

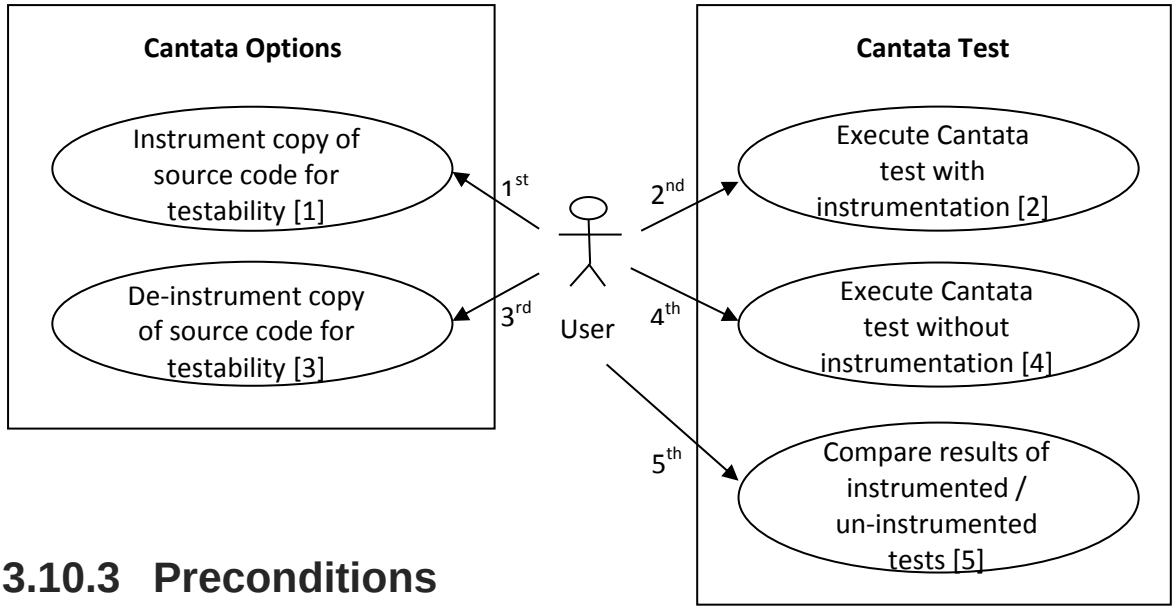
© Copyright QA Systems GmbH 2012

> That Source Code may contain data encapsulated as private in C++ or declared as static in C/C++, which cannot be verified without the test script having access to that data. In these cases Cantata Testability Instrumentation provides the ability to directly access and therefore verify data. Without this, such automated verification would not be possible, without using conditional compilation of the Source Code.

That Source Code may contain private methods or functions declared static which may only be indirectly verified through invoking the public / non-static methods or functions which call them. While not necessary to make the Source Code verifiable, this indirect verification approach is less efficient than isolation testing each function, and prone to one function masking an error in another function.

In these cases Cantata test script accessibility allows the tester to ensure that the Source Code can be verified, where the alternative is not even possible or is riskier than using test script access to the Source Code.

> That it may not be desired for source code to be altered in order to test it, implying that to alter it using Cantata testability instrumentation to provide the test script white-box access to source code may not be acceptable. If test script access instrumentation is used in the compilation of tests, it is not possible to build or run tests without that instrumentation.



3.10.3 Preconditions

Acceptable safety use scenarios for testability instrumentation have been documented and justified in the context of each test.

The Cantata Log level must be set to normal or higher for recording of full details of checks in the `<test_name>.CTR` files.

© Copyright QA Systems GmbH 2012

3.10.4 Use case descriptions

[1] Cantata testability instrumentation options must be set in the Cantata options file (ipg.cop) for the project. Testability instrumentation options may be set via the user interface in:

- Cantata Eclipse Workspace Code Generation Preferences,
- Eclipse Project Properties
- Cantata Test Script Manager

as defined in the Cantata User Interface Guide section of the Cantata Manual, or by manually editing the ipg.cop file(s) directly.

[2] Instrumented source code must be used in the build and running of the Cantata test executable as defined in the Testability Instrumentation section of the Cantata Manual.

[3] Testability instrumentation options may be separately enabled/disabled for each run of a Cantata test using Cantata project options.

[4] The non-instrumented code must be used in the build and running of the Cantata test executable.

[5] The results of the same test executed both with and without testability instrumentation must be compared to demonstrate that the instrumentation has not inadvertently affected the test result.

Where testability instrumentation has not been used to modify execution flow of the software under test, but only to check the actual against expected behaviour; the absence of checks implemented using any of the techniques below, must be the only differences between the test runs.

- Checks within Cantata wrappers on actual input and output parameter values passed between functions
- Checks within Cantata isolates on actual input parameters to functions isolated in the test
- Checks within Cantata wrappers or isolates on global data at that function call
- Checks within a Cantata test case using access variables for static data (C code) or automated friend declarations for private data (C++ Code)

Where testability instrumentation has been used to modify execution flow of the software under test (e.g. to inject failure conditions), any differences in the test results between instrumented and non-instrumented runs must be clearly documented and justified.

3.10.5 Constraints

None.

3.10.6 Dependencies

None.

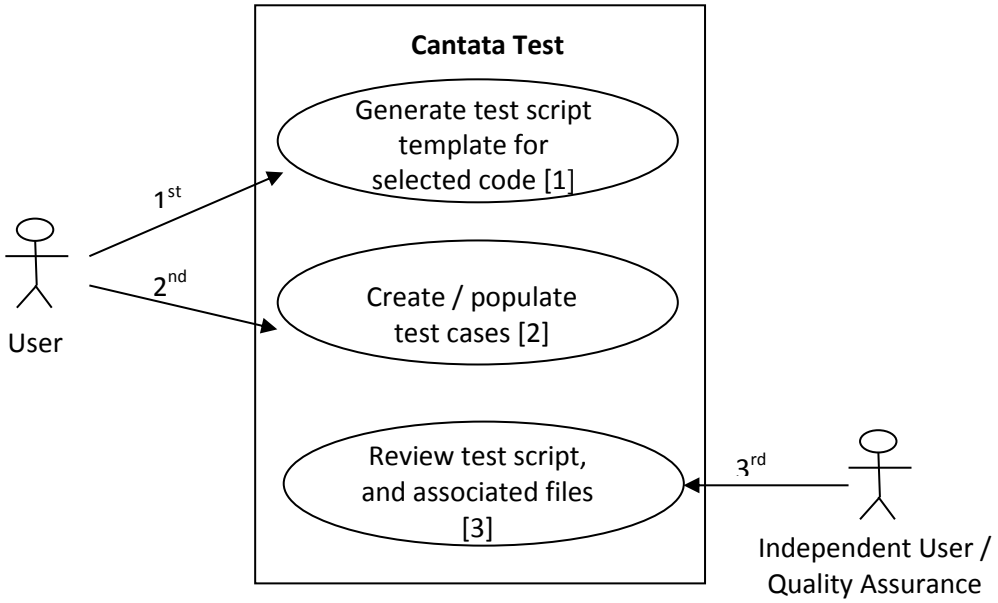
© Copyright QA Systems GmbH 2012

3.10.7 Post conditions

In a safety related context only the Cantata ASCII text `<test_name>.CTR` results files produced by the Core Product direct from the target with and without testability instrumentation must be used as certification evidence.

3.11 Creation of Cantata tests

Cantata test scripts are written as C or C++ source files that are linked with the source code for the software under test (SUT), the Cantata libraries and any other libraries required by the SUT to produce a "test executable".



3.11.1 Preconditions

Verification objectives for the test must exist.

A Cantata registered deployment for the target platform (both host and target if coverage reporting is being done on the host) must be available.

The Cantata Log for the Cantata coverage test level must be set to normal or higher for recording of full details of checks in the `<test_name>.CTR` files.

3.11.2 Use case descriptions

[1] A Cantata test script template may be created for selected source files / projects using the Cantata Test Script Manager Ancillary Tool as defined in the User Interface Guide section of the Cantata Manual, or manually in any code editor. A Cantata test script must conform to the layout restrictions defined in the User Guide Section of the Cantata Manual. Cantata will record a `SCRIPT_ERROR` where directives are not used in the correct order or manner.

© Copyright QA Systems GmbH 2012

[2] Cantata Test cases may be added to test scripts using the Cantata Test Script Manager Ancillary Tool as defined in the User Interface Guide section of the Cantata Manual, or manually in ay code editor.

[3] A Cantata test script, and any associated test cases data files (stored as header files), must be independently reviewed along with the Cantata project options, to ensure that they implement the verification objectives appropriate to the SIL and quality plan. Records of the test review must be maintained.

3.11.3 Constraints

The Cantata test must compile and link with the software under test to build a test executable.

3.11.4 Dependencies

None.

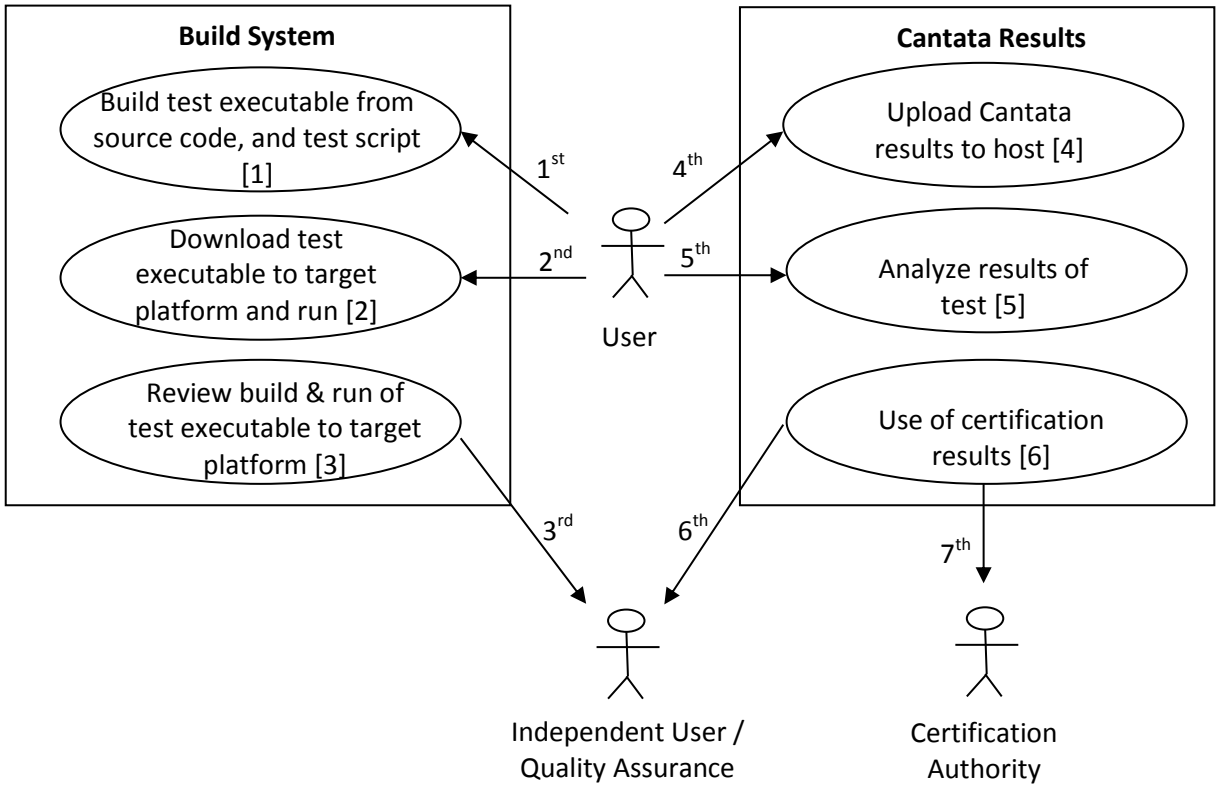
3.11.5 Post conditions

A reviewed Cantata test script and any associated header files for the source code under test.

© Copyright QA Systems GmbH 2012

3.12 Execution and Analysis of Cantata tests

Cantata Tests are built as test executables to run on the required target platform.



3.12.1 Preconditions

Verification objectives for the test must exist.

A Cantata registered deployment for the target platform (both host and target if coverage reporting is being done on the host) must be available.

The Cantata Log for the Cantata coverage test level must be set to normal or higher for recording of full details of checks in the `<test_name>.CTR` files.

3.12.2 Use Case descriptions

[1] The source code under test and the Cantata test script must be compiled and linked as a test executable for the target platform using one of the following mechanisms as defined in the User Guide section of the Cantata Manual:

- Cantata Makefiles via the Cantata Eclipse UI or on the CLI
- Users own build system with the Cantata compiler driver CPPCCD.
- Wind River Workbench Build Targets via the Workbench UI or headlessly on the CLI

© Copyright QA Systems GmbH 2012

Where a registered target deployment has been supplied with a platform specific technical note defining instructions required to correctly build the Cantata test executable on the host platform, this document must be a controlled Cantata artefact and the instructions followed.

[2] The built test executable must be downloaded to and executed on the target platform, using one of the following mechanisms as defined in the User Guide section of the Cantata Manual:

- Cantata Makefiles via the Cantata Eclipse UI or on the CLI
- Users own build system with the Cantata compiler driver CPPCCD
- Wind River Workbench Build Targets via the Workbench UI or headlessly on the CLI

Where a registered target deployment has been supplied with a platform specific technical note defining instructions required to correctly download and run the Cantata test executable on the target platform, this document must be a controlled Cantata artefact and the instructions followed.

[3] The commands for building and running of the Cantata test executable on the target must be independently reviewed.

[4] The Cantata registered deployment will have determined the I/O mechanism by which results (textual output) are written by the test executable and uploaded from the target platform to the host platform. The Cantata results are stored in the following files:

- `<test_name.CTR>` ASCII text results file
- `<test_name.CTG>` graphical results file for Ancillary Tool diagnostics
- `<test_name.COV>` coverage data results file

If file I/O is supported then this must be used to create the above Cantata results files. If file I/O is not supported then multiplexed output must be used on the target, and de-multiplexed on the host platform as defined in the Multiplexed Output section of the Cantata Porting Guide.

[5] The Cantata Eclipse user interface Ancillary Tools, may be used as detailed diagnostic aids, as defined in the Cantata User Interface Guide section of the Cantata Manual.

The certification evidence of the test result produced by the Cantata Core Product components directly from the target is only contained in the .CTR test results file which presents the results in a structured ASCII file.

Analysis of the .CTR file must include:

- Achievement of all test objectives,
- Where Cantata `CHECK_WARN` directives are used, that these do not result in a test failing,
- Absence of anomalies marked as any of the following:

© Copyright QA Systems GmbH 2012

- `>> FAILED`
- `>> Script Error`
- `>> NOT EXPECTED`
- `>> Unexpected Exception`

[6] The test build / execution mechanism commands for each Cantata test must be independently reviewed.

A configuration control system must be used to version control the test build / execution mechanism for each Cantata test.

The test build / execution mechanism must be provided to a certification authority as evidence of the means by which each Cantata test was undertaken.

3.12.3 Constraints

Where a registered target deployment records any limitations of the Cantata capabilities imposed by that target platform (e.g. lack of timing functions to perform timing analysis), the verification objectives for any test undertaking using that Cantata target deployment, must recognise such constraints and alternative verification strategies employed.

3.12.4 Dependencies

None.

3.12.5 Post conditions

In a safety related context only the Cantata ASCII text `<test_name>.CTR` results files produced by the Core Product direct from the target (both with and without instrumentation), together with the Cantata project options and test build / execution mechanism, must be used as certification evidence of the verification activity undertaken using Cantata.

© Copyright QA Systems GmbH 2012